VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Synchronizace adresáře s Google Drive

Folder Syncing with Google Drive

2013 Petr Zlotý

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Petr Zlotý**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Synchronizace adresáře s Google Drive
Folder Syncing with Google Drive

Zásady pro vypracování:

Cílem práce je vytvořit aplikaci, která bude synchronizovat obsah zvolené složky v operačním systému s účtem Google Drive mezi různými počítači. Pro tvorbu aplikace využijte Google Api. Implementace by měla být realizována v C++ nebo nad platformou .Net.

Obsah práce:
1. Zmapování situace na poli synchronizačních nástrojů pro cloud úložiště.
2. Popis Google API potřebné pro implementaci projektu.
3. Popis funkcionality, která musí být realizována.
4. Implementaci aplikace.
5. Testování aplikace.

Seznam doporučené odborné literatury:

Ján Hanák: Praktické objektové programování v jazyce C# 4.0, 978-80-87017-07-4, Artax 2009
Ján Hanák: Praktické paralelní programovaní v C# 4.0 a C++, 978-80-87017-06-7, Artax, 2009
Dále dle pokynů vedoucího práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Platoš, Ph.D.**

Datum zadání: 18.11.2011
Datum odevzdání: 07.05.2013

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky OSTRAVA

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 29.4.2013 .....

Poděkování

Rád bych poděkoval vedoucímu bakalářské práce Ing. Janu Platošovi, Ph.D. za odbornou pomoc a konzultaci při vytváření této práce.

Abstrakt

Cílem této bakalářské práce je implementace aplikace synchronizující adresář s cloudovým úložištěm Google Drive, analýza využívaného Google API a zmapování situace na poli synchronizačních nástrojů pro cloudové úložiště. Mezi hlavní požadavky na implementovanou aplikaci patří její vývoj nad platformou Microsoft .NET, využití .NET knihoven pro Google Data API, synchronizace uživatelem zvoleného adresáře bez nutnosti přesunu dat do zvláštní složky a to v rámci jednoho uživatelského účtu, mezi více zařízeními. Implementovaná aplikace umožní ochranu heslem a šifrování souborů pro zajištění nečitelnosti dat v datovém úložišti.

Klíčová slova:

Synchronizace, adresář, Cloud, Google Drive, Google Data API, Microsoft .NET Framework, .NET knihovny Google Data API

Abstract

The aim of this bachelor thesis is to implement application that sync directory with Google Drive cloud storage, analysis of used Google API and mapping the situation of synchronization tools for cloud storage. The major requirements of the implemented application are development with using the Microsoft .NET platform, using .NET client library for the Google Data API, syncing user-chosen directory without need to move data in separate folder within a single user account between multiple devices. The implemented application will allows password protection and encryption of files to provide unreadable data in storage.

Key words:

Synchronization, directory, Cloud, Google Drive, Google Data API, Microsoft .NET Framework, .NET library for the Google Data API

Seznam použitých symbolů a zkratek

API	– Application programming interface, rozhraní pro programování aplikací
GB	– Gigabajt, jednotka dat
GData	– Google Data
GDrive	– Google Drive
GMail	– Google Mail
ID	– Identifikace
MB	– Megabajt, jednotka dat
MS	– Microsoft
OS	– Operační systém
P	– Označení parametru metody
SDK	– Software Development Kit, sada vývojových nástrojů
sync	– Zkrácený výraz pro synchronizaci
ToS	– Terms of Service, podmínky užití
URI	– Jednoznačný identifikátor zdroje složený z URL a URN
URL	– Identifikace cesty ke zdroji
URN	– Identifikace zdroje
Win Phone	– Windows Phone
XML	– Extensible markup language, rozšířitelný značkovací jazyk
<T>	– Označení generických datových struktur

Seznam obrázků

Seznam výpisů zdrojového kódu

Obsah

1 Úvod

Cloudové služby jsou dnes velice rozšířeným trendem online uchovávání dat. Přístup k datům z různých míst a zařízení je dnes běžnou součástí, stejně tak požadavky na jejich sdílení. Cloud tyto možnosti nabízí. Existuje celá řada služeb nabízející nástroje a možnosti cloudové synchronizace. Právě analýzou těchto nástrojů a obecně využíváním cloudových služeb se bude tato práce zabývat.

V práci je řešeno, co vlastně Cloud uživateli nabízí a jak jej poskytují různé služby na poli Cloud computing. Jsou analyzované aspekty, které mohou vést při rozhodování o využívání služeb. Ať už se jedná o obecné výhody uchovávání dat mimo lokální a přenosná média, nebo potřeby zálohování či synchronizace právě dat lokálních zařízení. Nabízejí se ale i otázky bezpečnosti takto zpracovávaných a uchovávaných dat. Nejen jejich ochrana v online úložišti, ale i ochrana práv uživatele v různých situacích, které jsou v kapitole práce rozebrány.

Každá služba na poli cloudové synchronizace nabízí vlastní nástroj k využívání Cloudu. Jedná se nejčastěji o webové aplikace, ale také aplikační klienty pro lokální zařízení. Ostatní nabízející možnosti se celkově liší. Rozdíly mohou být v poskytovaném úložišti, v různých principech zpracovávání dat, ale i způsobech zabezpečení. Veškeré rozdíly mezi předními poskytovateli jsou analyzovány a zhodnoceny.

Hlavním cílem práce je návrh vlastní aplikace k řešení synchronizace lokálních dat s online úložištěm. Mezi stanovené cíle této aplikace patří synchronizace uživatelem vybraného adresáře mezi více zařízeními, za využití služby Google Drive. K tomu je nutné využívat veřejné API této služby. Základem je využití Microsoft .NET platformy pro implementaci aplikace, z toho plyne využívání Google API pro .NET.

Při vlastním řešení je nahlíženo na existující nástroje a jejich způsoby zpracovávání dat. S možnostmi Google API pro .NET jsou popsány všechna dosažená kritéria implementované aplikace a konečné zhodnocení implementačních možností a omezení, která vychází z volby využívané služby.

2 Synchronizační nástroje pro Cloud úložiště

2.1 Obecně o cloudových úložištích

2.1.1 Proč volit Cloud

V dnešní době stále častěji potřebujeme přistupovat ke svým datům z mnoha různých zařízení odkudkoliv z Internetu, z počítačů či mobilních zařízení. Potřebujeme synchronizovat a nejjednodušší způsob, jak toho docílit, je používat cloudové služby. Cloud není jen datové úložiště na serveru, ale nabízí možnosti sdílení dat mezi uživateli, poskytuje služby a aplikace Internetu. Nejedná se o novinku na poli poskytování internetových služeb, běžně využíváme Cloud v případě e-mailu či souborových serverů. S odstupem limitovaných velikostí schránek a rostoucí rychlosti internetového připojení se rozšiřují i technologické možnosti internetových aplikací. Důsledek je nárůst cloudových služeb.

Zřízením online úložiště dostanou uživatelé k dispozici datový prostor, který se využívá k ukládání dat. Může se jednat o soubory, dokumenty, fotografie, zálohy a různá data. Přistupovat k úložišti můžou uživatelé přes desktopové nebo mobilní zařízení, nebo z webových aplikací, znázorněno obrázkem 1[1]. Naprostá většina internetových úložišť nabízí ke stažení aplikace, pomocí kterých můžeme k datům jednoduše přistupovat, synchronizovat je, případně sdílet s dalšími uživateli. [1]

Obrázek 1: Cloud Computing

2.1.2 Užívání cloudových úložišť

Mezi základní využívací principy cloudových úložišť patří synchronizace dat. Ta probíhá u běžných cloudových nástrojů vytvořením jedné hlavní složky na lokálním disku uživatele a veškeré soubory umísťované do této složky se synchronizují s úložištěm. Jakákoliv změna dat na straně zařízení uživatele se projeví změnou dat úložiště a stejně tak opačně.

[1] http://www.qiktechnology.com/tag/cloud-computing

Některé služby umožňují volbu nastavení synchronizace konkrétních složek úložiště. Rozšířenými možnostmi synchronizace disponuje pouze malá část nástrojů. Některé služby nepotřebují data umísťovat do jedné hlavní složky, ale nabízí možnosti výběrů složek ze zařízení uživatele, které se současně synchronizují s úložištěm.

Data se na úložiště ukládají do stejné adresářové struktury, jako je struktura v hlavní složce na lokálním zařízení uživatele. Některé služby nepodporují adresářovou strukturu a data jsou ukládána do kořenového adresáře úložiště, povědomí o adresářové struktuře je obvykle uloženo v dalších souborech s metadaty.

Další využití cloudových úložišť je k zálohování dat. Naprostá většina služeb nabízí zálohování, avšak principy se liší. Je nutné, aby služba nabízela historii verzí souborů a to co největší.

Sdílení dat úložiště mezi více uživateli je standardem skoro každého úložiště. Principy jsou odlišné, od předávání veřejného odkazu na data po sdílení pouze mezi určitými zařízeními. Několik služeb zakládajících si na vyšší bezpečnosti dat a soukromí uživatele nepodporuje sdílení, nebo pouze nabízí část úložiště jako veřejný prostor, kde je nutné sdílená data přesunout. Důvodem je šifrování dat a ochrana heslem, tudíž jejich nečitelnost na straně úložiště, taková data je nemožno sdílet bez znalosti hesla.

Každá služba nabízí zdarma základní prostor cloudového úložiště pro uživatele. Velikost prostoru je v řádech od 2 GB po 7 GB v závislosti na službě. Placená úložiště nabízí větší prostor a obvykle se předplácejí na měsíční lhůtu. Ceny se liší u každé konkurenční služby.

Podpora všech platforem není u každé služby podmínkou. Vždy existuje u každé služby platforma či zařízení, které není úložištěm podporováno. Toto je ale vyvažováno existencí veřejného API každé služby a tak je k dispozici mnoho neoficiálních nástrojů a projektů třetích stran aby se doplnily mezery v podpoře.

2.1.3 **Aspekty při rozhodování pro cloudové úložiště**

Jako každá internetová služba či aplikace, jsou cloudové služby závislé na kvalitním internetovém připojení. Práce s daty bude vždy rychlá a efektivní pouze v závislosti na internetovém připojení. V případě nedostupnosti připojení k internetu během synchronizace, data se stanou neaktuální na jedné ze stran, nebo dochází k jejich ztrátě. Musí být zachovávána integrita dat. Nutnost je přístupnost klientské aplikace nebo webového rozhraní ze všech míst, kde se uživatel rozhoduje přistupovat k úložišti a manipulovat s daty.

Dalším aspektem na poli cloudových úložišť je závislost na poskytovateli. Pokud se uživatel rozhodne pro využívání služeb konkrétního poskytovatele, musí používat pouze jím určený software a je vystaven nutnosti se podřídit podmínkám poskytovatele. Tyto podmínky jsou odlišné pro každého poskytovatele. Výběr poskytovatele se může lišit podle následujících kritérií:

- Charakter ukládaných dat
- Ochrana osobních údajů
- Bezpečnost a spolehlivost
- Způsoby manipulace s daty
- Velikost úložného prostoru
- Platforma
- Osobní preference

Uživatel je dále závislý na schopnostech poskytovatele rozšířit úložiště podle budoucích potřeb uživatele. Důležitost navyšování kapacity úložiště až k možnostem neomezeného prostoru je také klíčový prvek při výběru. Uživatel je v neposlední řadě opět závislý na patřičném přístupu poskytovatele,

v případě změn, týkající se úložiště, či v případě řešení problémů vzniklých při porušení podmínek užívání služeb, více v podkapitole 2.1.4.

2.1.4 Ochrana osobních údajů a bezpečnost dat na serveru

Jedná se o nejvíce kritizovanou oblast na poli Cloud computingu. Ačkoliv z mého pohledu jsou některé kritizované sektory zásadní, jiné považuji za čistě individuální a to aspekty týkající se definice *citlivá a důvěrná data*. Je na každém uživateli, aby zvážil, jaká data pro něj tento pojem označuje a podle tohoto uvážení se dále rozhodoval o využívání cloudových služeb. Otázky bezpečnosti a samotná tato problematika bude vždy závislá na individuálních požadavcích uživatelů, při jejich konkrétních způsobech využívání cloudových úložišť.

Důležitou roli hraje seznámení s podmínkami používání a soukromí (ToS) poskytovatele úložiště. Každá služba má jiné podmínky využívání, díky tomu je možno pokrýt různé požadavky uživatelů, pro které je ochrana osobních údajů a bezpečnost dat prioritou při volbě úložiště.

Je nutné znát již zmíněný charakter dat, dále znát účel ukládání dat - a to sdílení, zálohování nebo synchronizace. Od těchto aspektů se odvíjí absence či nutnost vyšší bezpečnosti a šifrování souborů. Pouze v případě citlivých a důvěrných dat považuji za klíčové tyto rizika:

1) I když jsou data uložena na serveru šifrovaná, tak nemají šifrovaný obsah v úložišti. Data jsou běžně čitelná pro analýzu dat, pro náhledy a operace se soubory ze strany úložiště.

2) Spolehlivost a bezpečnost klientské aplikace nebo webového nástroje. Nástroj pro cloudovou synchronizaci by měl kromě bezpečnostních prvků využívat i šifrování a dešifrování dat pouze na lokálním disku uživatele a ukládat již šifrovaná data bez nutnosti uchovávat povědomí o dešifrovacích informacích na úložišti. Tímto bezpečnostním prvkem se ale omezuje sdílení dat mezi uživateli bez dešifrovacích nástrojů a bez znalosti dešifrovacích informací.

3) Data mohou být poskytovatelem úložiště poskytnuta třetí straně mimo vědomí uživatele. Poskytovatel může být podřízen jurisdikci, která mu uděluje povinnost předávat data uživatele vládním organizacím. Existují především rozdílné legislativní pohledy na ochranu osobních údajů a dat jednotlivce mezi USA a Evropskou unií. [2]

4) Analýza obsahu dat ze strany poskytovatele, z důvodu kontroly před porušováním práv při ukládání dat porušujících podmínky využívání služeb. Je důležité, jak s takovým obsahem poskytovatel bude nakládat. Zde je možná ztráta dat, zablokování účtu a to z vědomého, ale i nevědomého (chybného) počínání uživatele.

2.2 Služba Dropbox

2.2.1 Popis služby

Dropbox je často využívaná služba cloudové synchronizace mezi naprostou většinou operačních systémů a mobilních zařízení. Existuje velké množství aplikací třetích stran díky veřejnému API. Vyznačuje jednoduchostí a přístupností. Podporuje sdílení vlastních dat s ostatními uživateli, je možné přistupovat k úložišti ze sociálních sítí ke sdílení multimédií a nabízí jisté klientské výhody registrovaným uživatelům. [1]

2.2.2 Principy synchronizace

Synchronizace funguje na principu pouze jedné hlavní složky na lokálním disku klienta a data umístěná v této složce se synchronizují s cloudovým úložištěm. Naopak je možné v Dropbox úložišti tvořit adresářové struktury dle libosti.

2.2.3 Možnosti úložiště

Aktuálně Dropbox nabízí registrovanému uživateli základní 2 GB prostor. Standardem je dokoupení většího prostoru, při srovnání s konkurenčními službami, stojí relativně více. Neexistují zde limity na velikosti ukládaných souborů a je možné získat odměny stylem navyšování prostoru úložiště o 500MB při doporučení novému uživateli k registraci Dropbox účtu. Maximální placený prostor je 500 GB.

2.2.4 Ochrana dat

Autentizační údaje uživatel zadává pouze při nastavení klientské aplikace. Pro přístup k úložišti se využívá pouze ID zařízení. Je jednoduché vyhledat informace, jak získat toto ID uložené v souboru na zařízení uživatele a zneužít ho k přístupu k úložišti. Bezpečnostní riziko tento čin nepředstavuje, protože k získání ID je nutný přístup k zařízení uživatele a v tomto případě existuje i přímý přístup k lokálním datům. Problém nastává díky pevnému ID, které se nemění. Právě zde existuje největší bezpečnostní riziko pozdějšího zneužití jakýchkoliv dat, které v budoucnu napadený uživatel uloží do úložiště.

2.2.5 Veřejné API

Dropbox nabízí SDK pro Sync API a Core API. Dropbox Sync API SDK umožní přidávat klienta Dropbox k vlastním aplikacím. Možnosti pouze pro iOS a Android. Dropbox Core API SDK umožní tvorbu vlastních aplikací za využití metod pro čtení a zápis, sdílení, vyhledávání a práci se soubory DropBox úložiště. [6] Existují SDK pro iOS, Android, Python, Ruby, Java, OS X, .NET a PHP jako SDK třetích stran

2.2.6 Shrnutí služby

- Služba disponuje přehledným a jednoduchým rozhraním.
- Podporuje mnoho platforem.
- Nabízí klientské výhody získáváním nových uživatelů.
- Základní prostor úložiště je pouze 2 GB a vyšší ceny oproti konkurenci.
- Nenabízí ochranu dat heslem a jejich šifrování.
- Uživatel si musí klást větší důraz na ochranu zařízení před nevyžádaným přístupem.

2.3 Služba Microsoft SkyDrive

2.3.1 Popis služby

SkyDrive je cloudová služba společnosti Microsoft, z toho plynou hlavní výhody pouze především pro uživatele operačních systémů MS Windows. Microsoft se snaží o prvenství v oblasti osobního cloudu, to dokládá parametry úložiště, které jsou oproti konkurenci lepší. Dále i neustálými změnami webového rozhraní a zvyšováním rychlosti služeb. [1]

2.3.2 Principy synchronizace

Synchronizace probíhá pomocí jedné složky na uživatelském zařízení. Data je možné stahovat do podporovaných mobilních zařízení. V OS Microsoft Windows je možné přesměrovat složky do složky SkyDrive a synchronizovat tak více složek najednou.

2.3.3 Možnosti úložiště

SkyDrive nabízí pro uživatele jako základní úložiště kapacitu 7 GB, je zde pouze omezení velikosti ukládaného souboru na 2 GB. Maximální placený prostor je 100 GB.

2.3.4 Ochrana dat

Je nutné dobře znát pravidla užívání Microsoft SkyDrive cloudového úložiště. Jsou dána pravidla, která určují závadný nebo nevhodný obsah v tomto úložišti a v případě porušení těchto pravidel se uživatel vystavuje riziku zrušení účtu ze strany Microsoft a ztrátě dat. Neexistuje totiž povinnost poskytovatele vydat data zablokovaného účtu. [4]

2.3.5 Veřejné API

Microsoft nabízí Live Connect API, díky kterému je možnost vytvářet aplikace pracující s SkyDrive složkami, soubory a multimédii. Pro usnadnění práce s API existují Interactive Live SDK nástroje. Implementace je možná v JavaScript, C#, Objective-C, REST a Java.

2.3.6 Shrnutí služby

- Nabízí velký základní prostor úložiště o kapacitě 7 GB.
- Microsoft nabízí příjemné ceny rozšiřování úložiště, avšak jen do kapacity 100 GB.
- Absence ochrany konkrétních dat heslem.
- Podporuje pouze určité platformy.
- Nutnost přesně znát definice nevhodného či závadného obsahu z pravidel užívání.

2.4 Služba Google Drive

2.4.1 Popis služby

Jedná se o nástupce služby Google Documents, webové rozhraní připomíná tuto původní službu, ale služba samotná je jen jejím dalším vývojovým stupněm s jistými novinkami a různým vylepšením, které Google Drive řadí mezi běžné cloudové služby. Výhody využívání této služby přivítají převážně uživatelé aplikací Google. Klient aplikace pro synchronizaci je dostupný pouze pro Windows a OS X a mobilní zařízení se systémem Android. Veřejné API umožňuje tvorbu neoficiálních klientů pro ostatní platformy ale i stávající.

2.4.2 Principy synchronizace

Instalace klienta na zařízení uživatele vytvoří složku na disku, s kterou se synchronizují data z úložiště. Neexistuje zde vzájemná synchronizace více složek, v nastavení klientské aplikace může uživatel volit podložky, které se budou synchronizovat s podsložkami úložiště. Je možná adresářová struktura na úložišti oproti původním Google Documents.

2.4.3 Možnosti úložiště

Google poskytuje jako úložiště konkrétní plány. Kapacita úložiště je sdílená mezi službami Google Drive a Google+ Photos. Současně plán definuje kapacitu pro úložiště e-mailů Gmail, které je oddělené od úložiště Google Drive. Základní úložiště zdarma poskytuje 5 GB prostor pro nekonvertované soubory. Je možné dokoupit plány od 25 GB po 16 TB.

Úložiště dále nabízí původní funkcionalitu Google Documents vytváření dokumentů v nativním formátu Google Documents a práci s nimi přímo z webového rozhraní a možnosti jejich sdílení. [1]

2.4.4 Ochrana dat

Google Drive neposkytuje ochranu dat heslem. V zásadách ochrany osobních údajů a ochrany dat Google Apps, mezi které patří Google Drive, jsou uvedeny všechny informace ohledně zabezpečení a nakládání s daty. [5]

2.4.5 Veřejné API

Google Drive SDK nabízí knihovny pro práci s úložištěm, ukládání a stahování souborů. Nabízí ale i více než jen práci s úložištěm a integraci aplikací s Google Drive. SDK nabízí knihovny pro Python a Objective-C. Dále nabízí knihovny, které jsou ve fázi ve vývoje, ale využitelné s jistými limity:

- Knihovny pro .NET, Java, JavaSkript a PHP

2.4.6 Shrnutí služby

- Google Drive sází na jednoduchost jak aplikačních klientů, tak sdílení.
- Výhodou je tvorba dokumentů nativního formátu a jejich sdílení.
- Google jako jeden z mála nabízí úložiště pro svou službu o kapacitě do 16 TB.

2.5 Služba SpiderOak

2.5.1 Popis služby

Volbu tohoto úložiště přivítají uživatelé, jejichž osobní požadavky využívání cloudových úložišť jsou zaměřeny na zálohování dat. Při použití desktopového klienta služba nabízí šifrování a ochranu dat heslem a odpadá těmto datům možnost veřejného sdílení a přístup z webového klienta. Sdílení je umožněno z oddělených složek, ve kterých uživatel definuje viditelnost obsahu.

2.5.2 Principy synchronizace

Synchronizace probíhá jako u běžných cloudových nástrojů. Je možné vybírat složky, které budou synchronizovány s úložištěm.

2.5.3 Možnosti úložiště

Služba nabízí základní úložiště zdarma o velikosti 2 GB, placené úložiště lze navyšovat do 1 TB.

2.5.4 Ochrana dat

SpiderOak je ideální řešení pro uživatele žádající větší zabezpečení dat, neboť služba nabízí šifrování dat a jejich ochranu heslem. Heslo k šifrování souborů uživatel zadává pouze na svém zařízení a pouze zde jsou data šifrovaná a heslo se neukládá do úložiště společně se soubory. Názvy souborů na serveru nejsou známy, proto tyto akce jsou prováděny pouze přes desktopového klienta služby. Každopádně

dodatečná ochrana dat je velice výhodná a nedisponuje tímto faktorem každá služba cloudové synchronizace.

2.5.5 Veřejné API

Služba vydává GPLv3 knihovny pro integraci možností úložiště do aplikací uživatelů. Oficiální knihovny jsou programovány pouze v jazyce Python.

2.5.6 Shrnutí služby

- Mezi velkou výhodu patří u této služby šifrování a ochrana dat heslem.
- Uživatelům hledajícím komplexní zálohovací nástroj nabízí rozšířené možnosti.
- Aplikační klient nabízí synchronizování více vybraných složek současně.
- Služba nabízí jako základní úložiště pouze 2 GB prostor pro data.
- Díky rozšířeným možnostem ale ztrácí aplikační klient na jednoduchosti ovládání.

2.6 Služba Wuala

2.6.1 Popis služby

Služba Wuala je dalším z malé řady služeb, které ocení uživatelé jejichž prioritou při výběru cloudových služeb je bezpečnost a vyšší ochrana dat. Služba je multtiplatformní, nabízí šifrování a ochranu dat heslem, nabízí pouze aplikačního klienta.

2.6.2 Principy synchronizace

Způsob synchronizace se neliší od běžných synchronizačních nástrojů, provádí se pomocí hlavní složky klientské aplikace, služba disponuje i možnostmi automatického zálohování dat.

2.6.3 Možnosti úložiště

K dispozici je základní úložiště zdarma o velikosti 5 GB. Je možno kupovat dodatečná úložiště pouze na roční období do maximální velikosti 100 GB.

2.6.4 Ochrana dat

Služba Wuala si silně zakládá na bezpečnosti a ochraně dat uživatele. Neexistuje webové rozhraní, pouze klientská aplikace. Šifrování dat se provádí pouze v aplikaci před uložením dat na server, heslo použije uživatel pouze před šifrováním a úložiště nemá o heslu povědomí. Ani poskytovatel úložiště nemá povědomí o datech, neboť jsou šifrovaná a nečitelná, a dále se soubory rozdělují na části, které se ukládají na více serverů. Při zapomenutí hesla uživatelem, není možné získat data zpět. [3]

2.6.5 Veřejné API

Služba Wuala nenabízí veřejné API.

2.6.6 Shrnutí služby

- Služba Wuala klade vysoký důraz na bezpečnost dat uživatelů.
- Nabízí šifrování a ochranu dat heslem.
- Základní nabízené úložiště je běžné velikosti, avšak maximální pouze o kapacitě 100 GB.

3 Popis Google .NET API využívané aplikací

3.1 O použitém API

Implementovanou aplikací jsou využívány Documents List API knihovny, které poskytuje Google Data API. Aktuálně již je Google Data API označeno *deprecated*. Další vývoj byl ukončen a touto politikou označené API a knihovny Documents List je možné využívat do první poloviny roku 2015, kdy bude podpora úplně zastavena. [8] Odůvodnění využívání tohoto API je uvedené v kapitole 5.4.

Klientské .NET knihovny Google Data API jsou udržovány jako open-source Google Project. Knihovny jsou distribuovány MSI instalátorem, obsahujícím knihovny ke všem službám Google. [9] Pro dnešní Google aplikace existuje Google Apps Application APIs, do kterého spadají všechny API služeb Google.

Google Documents List API umožňuje klientské aplikaci stahovat a ukládat data, vytváření, úpravu a mazání souborů a složek úložiště. Toto je nutné k synchronizaci s cloudovým úložištěm. Všechny interakce s API vyžadují ověřování platnými uživatelskými Google účty. API rozlišuje následující základní pojmy:

- Document – nativní formát Google dokumentů, nejedná se o třídu Document
- File – běžné soubory ukládané do úložiště
- Collection – kolekce, což je označení pro složku úložiště
- Resource – každý prvek udržovaný API, jedná se o soubory a kolekce
- Resource ID – ID kterým jsou identifikovány veškeré prvky

Google Documents List API nabízí tyto základní aplikační možnosti pro implementaci synchronizační aplikace:

- Autorizační požadavky
- Získávání základních informací o uživatelském účtu
- Získávání změn prvků v úložišti
- Získávání seznamu dokumentů, souborů a kolekcí
- Vytváření dokumentů a kolekcí
- Ukládání dokumentů a souborů
- Stahování dokumentů a souborů
- Úpravy a změny dokumentů
- Odstraňování dokumentů a souborů a kolekcí

3.2 Knihovny potřebné pro běh aplikace

3.2.1 Analýza knihovny Google.GData.Client

Jedná se o hlavní jmenný prostor obsahující základní třídy. Pomocí těchto tříd je možnost pracovat s každým GData nastavením. Obsahuje přes 120 tříd a další děděné třídy v jiných knihovnách. Popis využívané funkcionality v sekci 3.3. Nejpodstatnější třídy pro implementovanou aplikaci jsou:

- Entry – Základní třída pro Feed třídy, které z ní dědí. Objekt reprezentuje prvek úložiště.
- Feed <T> – Generická třída. Pomocí property Entries se získá kolekce všech objektů úložiště.
- FeedRequest <T> – Základní třída pro objekty tvořící požadavky pro server.
- RequestSettings – Žádající třída, udržuje pověření (Credentials) a další nastavení, které je předáváno jiným objektům při požadavcích pro server.
- GDataCredentials – Objekt držící autentizační údaje.
- ClientLoginAuthenticator – Objekt držící autorizační údaje po úspěšné autentizaci, tyto informace jsou potřebné pro inicializaci ukládání souborů do úložiště.
- GDataRequestFactory – Základní třída sloužící k přetypování objektů při nastavování konkrétních vlastností DocumentsService.
- ResumableUploader – Třída poskytující protokol ukládání dat na úložiště.

- Document (děděné z Client.Entry, má vlastní jmenný prostor) – Základní třída pro veškeré prvky úložiště. Se získanými prvky z úložiště se dále pracuje jako s datovým typem Document. Prvek typu Document může být jak soubor, nativní dokument, tak i kolekce. Objekt drží všechny potřebné údaje o souboru (popř. kolekci), získávají se pomocí property:
    - Id – vrací ID prvku úložiště v podobě URI
    - Self – vrací ID prvku úložiště v podobě URI, ale s jiným URL
    - ParentFolders – vrací rodičovský prvek (kolekci), pokud se jedná o soubor kolekce
    - QuotaBytesUsed – vrací velikost zabírajícího prostoru prvkem
    - Type – poskytuje typ prvku jako file, folder, nebo typ nativního dokumentu
    - Title – poskytuje název prvku úložiště
    - Updated – vrací čas poslední manipulace s prvkem
    - ResourceId – poskytuje zkrácený tvar Id bez URL
    - DocumentEntry – vrací DocumentEntry z GData.Documents

  Třída je součástí knihovny Google.GData.Client a má vlastní jmenný prostor Google.Documents. Při každé integraci je nutné používat tento jmenný prostor.

- DocumentsRequest (děděné z Client.FeedRequest, má vlastní jmenný prostor) – Třída sloužící k získání prvků úložiště jako kolekci Feed<T>. Při inicializaci objektu DocumentsRequest musí být předán objekt RequestSettings s pověřením a kolekce Feed se získá pomocí metod:
    - GetEverything() – vrátí veškeré prvky úložiště a to soubory a kolekce
    - GetFolders() – vrátí pouze kolekce
    - GetFolderContent(p) – získává pouze prvky kolekce, která se předává jako parametr

  Metodou Download se stáhne požadovaný prvek předaný parametrem jako typ Document. Metodou Insert se vytvoří kolekce podle parametrů DocumentType jako Folder a URI [8]:
    - URI rozlišující kořenovou (root) kolekci, která je rodičovská pro všechny prvky
    - URI rozlišující kolekci, která je potomek (child), je umístěná v kořenové kolekci

Každý prvek má v ID obsaženo URL a URN. URL definuje možnosti umístění souborů v úložišti při jejích vytváření či odstraňování. URL je možné získávat pomocí třídy DocumentsListQuery.

URN slouží k identifikaci prvků, kompletní URI je složení obou, přičemž rozlišení kolekce je výrazem „folder:“ a rozlišení souboru výrazem „file:“ mezi oběma identifikátory, Společně dávající kompletní URI.

3.2.2 Analýza knihovny Google.GData.Documents

Jmenný prostor poskytující třídy pro programový přístup a manipulaci s daty úložiště. Mezi aplikací využívané třídy patří:

- DocumentsService – Je hlavní třídou k provádění požadavků na server. Použité metody:
  - Delete – odstranění kolekce z úložiště i s jejími prvky, nutné předat parametr jako Id existující kolekce formou URI
  - UploadFile – jednoduchá metoda k uložení souboru na server
  - Query – předává třídě objekt DocumentsListQuery
  - setUserCredentials – nastaví pověření třídě a ověří se autentizační údaje
- DocumentsListQuery – Slouží k vytváření URI dotazů podle různých typů souborů.

3.2.3 Analýza knihovny Google.GData.AccessControl

Tento jmenný prostor obsahuje elementy a metody pro zacházení s AccessControl seznamy. Tyto seznamy jsou využívány v případě zavádění autorizace. Třídou AclEntry se definují položky objektu AclFeed. Přistupuje se k položkám AclEntry pomocí property Entries. Třídou FeedQuery se vytváří URI AccessControl, které zpřístupní AclFeed pro Service.Query (z Google.GData.Client).

Třída FeedQuery je podtřída Google.GData.Client.FeedQuery. Jde vidět závislost této knihovny na knihovně Google.GData.Client. Využívaná funkcionalita této knihovny je při inicializaci DocumentsService objektu.

3.2.4 Analýza knihovny Google.GData.Extensions

Tento jmenný prostor zahrnuje elementy, které jsou používány napříč některými GData nastaveními.

3.3 Integrace funkcionality knihoven ve vlastní aplikaci

3.3.1 Využívání DocumentsRequest třídy

Třída DocumentsRequest se využívá k získání seznamu souborů z kolekce, k stáhnutí souborů, k odstranění souborů a vytváření kolekcí. Popsáno v kapitole 3.2.1. Při inicializaci DocumentsRequest je zapotřebí objektu třídy RequestSetting z Google.GData.Client. Jmenný prostor DocumentsRequest třídy je Google.Documents. Při využití třídy k získání seznamu prvků ze serveru, je potřeba instance třídy Feed<T> datového typu Document a provedení požadavku je ilustrováno v ukázce výpisu č. 1.

```
DocumentsRequest docRequest = new DocumentsRequest(requestSetting);
Feed<Document> feed = docRequest.GetEverything();
IEnumerable<Document> allServerItems = feed.Entries;
```

Výpis 1: Požadavek k získání všech prvků z úložiště.

Se získanými prvky se dále pracuje jako s objekty třídy Document, takto získané soubory a složky (resp. kolekce) mají všechny vlastnosti prvků z úložiště, popsány v kapitole 3.2.1. Jsou možnosti, jak získat pouze seznam kolekcí z úložiště, ilustrováno ukázkou č. 2.

```
DocumentsRequest docRequest = new DocumentsRequest(requestSetting);
Feed<Document> feed = docRequest.GetFolders ();
IEnumerable<Document> allServerCollections = feed.Entries;
```

Výpis 2: Požadavek k získání pouze kolekcí (složek) z úložiště.

```
DocumentsRequest docRequest = new DocumentsRequest(requestSetting);
Feed<Document> feed = docRequest.GetFolderContent(existingFolder);
IEnumerable<Document> allServerFolders = feed.Entries;
```

Výpis 3: Požadavek k získání souborů složky z úložiště.

Pro získání výhradně souborů konkrétní kolekce, je potřeba znát reprezentaci existující kolekce objektem Document a předat objekt metodě GetFolderContent. Ilustrováno ukázkou č. 3. Tento případ ukazuje situaci, kdy k získání konkrétních prvků z úložiště, se vyžadují už získané rodičovské (parent) prvky jako objekty Document. V případech stahování a odstraňování prvku z úložiště je nutné znát existující prvek jako objekt Document, při tvorbě prázdné kolekce tato nutnost odpadá:

```
DocumentsRequest docRequest = new DocumentsRequest(requestSetting);
Document doc = new Docuemnt();
Doc.Type = Google.Documents.Document.DocumentType.Folder;
Doc.Title = collectionName;      // proměnná s názvem kolekce
docRequest.Insert<Document>(new
URI(DocumentsListQuerry.documentsBaseUri), doc);
```

Výpis 4: Vytvoření prázdné kolekce v úložišti.

Ve výpisu 4, při vytváření kolekce se předává metodě URI, identifikující místo vytvoření prvku v úložišti. URL se tvoří pomocí třídy DocumentsListQuerry a označují místo vytvoření v úložišti. URL má tvar `https://docs.google.com/feeds/default/private/full` a definování kolekce je pomocí přidání výrazu `/folder%3A` a přiřazení identifikace prvku, což tvoří kompletní URI. Takto se vytváří Document.Self každé kolekce v úložišti. Rozdíl identifikace souborů je pouze ve výrazu `/file%3A`. Nutno podotknout rozdílné URL mezi Document.Id a Document.Self:

- Id musí mít URL [8]: `https://docs.google.com/feeds/id/`
- Self musí mít URL [8]: `https://docs.google.com/feeds/default/private/full/`
- URN je stejné pro obě property, s rozdílem identifikačních výrazů souborů nebo kolekcí.
- Lze získat pomocí Document.ResourceId, které poskytuje právě ono identifikační URN.

Případ stahování souborů z úložiště vyžaduje také DocumentsRequest třídu a platné pověření od RequestSettings objektu, současně se znalostí existujících prvků, jako objektů Document. Metoda Download vrací Stream poskytnutého souboru, který se dále zpracuje pomocí obvyklých postupů zpracování toku dat a vytváří soubor na určeném místě. K získání toku dat je možné využít také třídu DocumentsService. Použití metody Download v ukázce č. 5:

```
DocumentsRequest request = new DocumentsRequest(requestSetting);
//doc – soubor úložiště typu Document, exportFormat – formát souboru
using (Stream stream = request.Download(doc, exportFormat))
{
     //zpracování toku dat
}
```

Výpis 5: Použití metody Download k získání toku dat.

3.3.2 Využívání DocumentsService třídy

Třída DocumentService se využívá k vytváření, úpravě a odstraňování prvků úložiště, především dokumentů nativního formátu Google. Aplikace využívá tuto třídu k nastavení uživatelského pověření, odstraňování kolekcí a získávání přístupu ke zdrojům DocumentEntry ze třídy GData.Documents.

Odstraňování kolekce probíhá nejprve jejím vyhledáním v získaných seznamech kolekcí, nebo všech prvků z úložiště. Dále následují kroky:

- Využije se Document.ResourceId z jmenného prostoru Google.Documents pro získání identifikace existující kolekce v úložišti a přidá se URL k identifikaci umístění. Existují libovolné způsoby získávání identifikací existujících prvků, ovšem je nutné získat správné URI identifikující kolekci.
- Přidá se za URI parametr `?delete=true` a zavolá se metoda Delete s parametrem této URI.

Mimo uvedené kroky je možnost odstraňovat kolekce pomocí jejich výběru [8]:

```
DocumentsService service = new DocumentsService(appName);
FolderQuery query = new FolderQuery();
DocumentsFeed feed = service.Query(query);
if(feed.Entries.Count > 0)
{
     DocumentsEntry folder = (DocumentsEntry)feed.Entries[index];
     Service.Delete(folder, true);
}
```

Výpis 6: Odstranění kolekce z úložiště.

Při odstranění kolekce je možnost parametrem delete=true volit možnost jejího trvalého odstranění, opačně se pouze přesune do koše. Soubory, které kolekce obsahuje, se odstraní také. Při velkém množství souborů kolekce, se tyto soubory přesunují do kořenové kolekce a pomalu se odstraňují po blocích velikosti 50.

Odstraňování konkrétního souboru kolekce probíhá stejným způsobem jako uvedené příklady odstraňování s rozdílem těchto částí:

- Musí být zajištěno URI existujícího souboru.
- V případě ukázky č. 6. se místo FolderQuery použije DocumentsListQuery objekt.

Při požadavku na stahování souborů z úložiště je možné využívat DocumentsService třídu k získání

toku dat. Zde není zapotřebí znát typ souboru. Je zapotřebí znát pouze soubor jako Document a URI se získá pomocí DocumentEntry, ukázka v ukázce č. 7:

```
Uri uri = new Uri(doc.DocumentEntry.Content.AbsoluteUri)
using (Stream stream = service.Query(uri))
{
     //zpracování toku dat
}
```

Výpis 7: Využití DocumentsService k získání toku dat.

3.3.3 Využívání ResumableUploader třídy

ResumableUploader třída nabízí rozšířené možnosti ukládání souborů do úložiště ve srovnání s třídou DocumentsService. Nabízí včetně jednoduchého uložení i asynchronní ukládání, navazování a přerušení ukládání. Nabízí také dvě události:

- AsyncOperationProgress
- AsyncOperationCompleted

Volba místa uložení souborů v úložišti probíhá opět podle URL nebo URI, je nutné volit pro:

- Ukládání do kořenové kolekce (root). URL musí mít následující tvar:
  http://docs.google.com/feeds/upload/create-session/default/private/full/
- Ukládání do vlastní existující kolekce. V tomto případě je nutné použít URI identifikující tuto kolekci, místo výše uvedené URL.

Při využití obou událostí ResumableUploader je možné přesně zaznamenat průběh ukládání souborů na úložiště. AsyncOperationProgress předává procento aktuálního zpracování souboru, AsyncOperationCompleted oznamuje dokončení ukládání.

4 Návrh implementace

4.1 Požadované modely budoucího systému

Pro požadovanou funkcionalitu aplikace jsou stanoveny následující požadavky k implementaci:

- Přihlášení a odhlášení uživatele
- Komunikace s úložištěm
- Volba adresářů
- Ukládání souborů na úložiště
- Stahování souborů z úložiště
- Synchronizace dat

Uvedené modely jsou výsledkem analýzy návrhu funkcionality. Popisují z různých pohledů, co se bude aplikací provádět k zajištění požadavků funkcionality, uvedené v zadání práce.

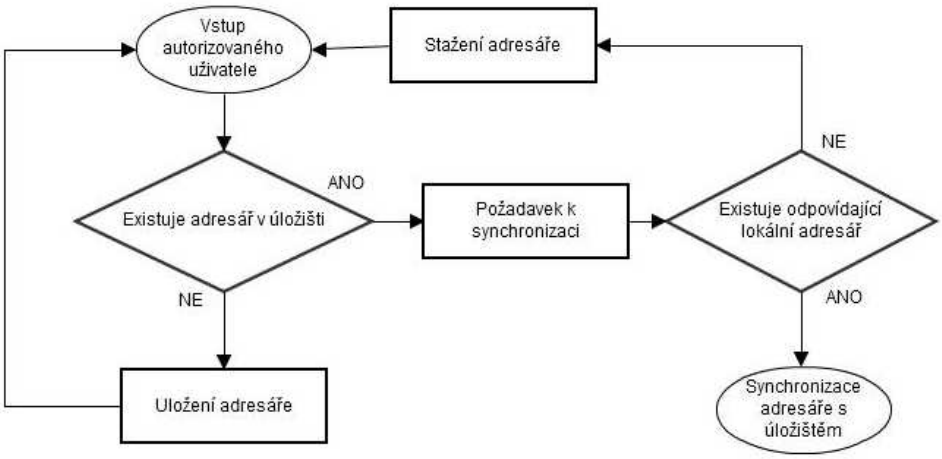
4.2 Návrh řešení a popis komunikace

Podrobný pohled uvádí data a funkce zpracovávané požadovanými modely a jejich komunikaci s okolím a ostatními modely.

Analýzou zvoleného API ke komunikaci s úložištěm, jsou následující výsledky potřebné k prioritnímu implementování:

- Autentizace uživatelem zadaných přihlašovacích údajů.
- Získávání seznamu všech prvků kolekce před operacemi s daty.

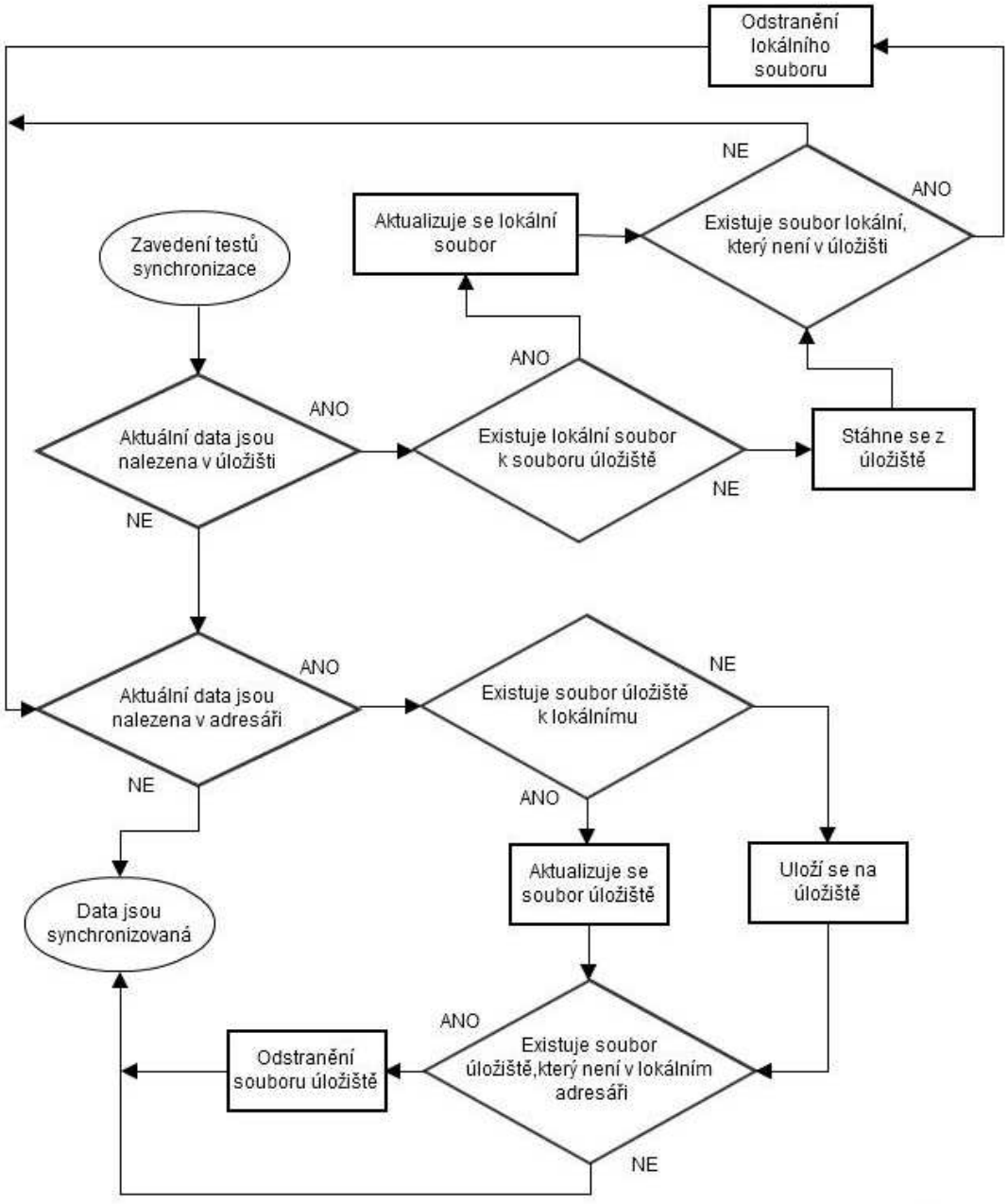
Následují operace, které jsou přímo závislé na těchto uvedených faktorech. Tyto operace vyžadují jednak existenci lokálního adresáře, tak kolekce v úložišti. Komunikace je znázorněná obrázkem 2:



Obrázek 2: Operace při volbě adresáře

Operace synchronizace budou závislé na konkrétních stavech dat. Analýzou možností úložiště bylo docíleno následujících aspektů, které je nutné implementovat. Komunikace zobrazena obrázkem 3.

- Testování aktuálních – nejnovějších dat kolekce úložiště a poté lokálního adresáře.
- Ukládání nebo stahování přidaných dat do kolekce oproti adresáři a stejným způsobem opačně.
- Odstraňování přebývajících – neaktuálních dat kolekce oproti adresáři a opačně.



Obrázek 3: Testování dat při synchronizaci

5 Popis implementované aplikace

5.1 Použité technologie a knihovny

Aplikace je implementována v jazyce C# nad .NET Framework 4. Pro vývoj bylo využito vývojového prostředí Microsoft Visual Studio 2010. Jako nestandartní knihovny jsou využity .NET knihovny Google Data API, popsané v kapitole 3. Pro upřesnění, jedná se o tyto knihovny:

- Google.GData.AccessControl
- GoogleGData.Client
- GoogleGData.Documents
- Google.GData.Extensions
- Ionic.Zip - volně šiřitelná knihovna DotNetZip, poskytuje funkcionalitu archivace
- Newtonsoft.Json - JSON framework pro .NET, k zpracovávání datového formátu JSON

5.2 Popis vlastních modulů a knihoven aplikace

Celé řešení obsahuje projekty knihoven tříd a projekt s Windows Forms komponenty pro uživatelské rozhraní. Jádro funkcionality aplikace a veškeré struktury jsou zapsány ve vlastních knihovnách, které jsou součástí hlavní složky aplikace.

5.2.1 Vlastní struktury a jejich sestavení (assembly)

Vlastní součásti aplikace jsou následovně uvedeny. Název jmenného prostoru odpovídá názvu sestavení, mimo aplikační. Většina tříd má více konstruktorů k zpracovávání rozdílných požadavků stejných nebo podobných operací, které třídy poskytují:

- **SynchGDrive.exe**
  Spustitelný soubor aplikace, definuje uživatelské rozhraní a zpracování funkcionality knihoven. Sestavení má jmenný prostor s názvem APP a obsahuje komponenty:
  - TaskbarApp – hlavní proces běžící na pozadí, udržuje instanci WorkWindow
  - WorkWindow – rozhraní aplikace, zpracovává funkcionalitu knihoven přes události
  - PasswordEntry – vstupní formulář pro ověření hesla pro šifrování
  - ProgressBox – komponenta zobrazující průběh během určitých operací
  - Options – okno nastavení vlastností aplikace
  - FilePreview – okno zobrazení náhledu XML souborů s daty
  - Accounts – okno nastavování uživatelských účtů pro rychlé přihlašování
  - Help – okno s informacemi o ovládání a používání aplikace
  - About – okno s popisem verze a sestavení aplikace

- **AppProcessing.dll**
  Jádro funkcionality aplikace. Veškeré funkce a algoritmy zpracovávání operací aplikace, kromě ukládání na úložiště, tvorby archívů, šifrování a zpracovávání konfiguračních souborů. Sestavení obsahuje třídy:

- Backup – třída zpracovává a vytváří zálohy lokálního adresáře před synchronizací
- CollectionManage – funkce pro tvorbu a odstraňování kolekcí v úložišti
- DirectoryBrowser – poskytuje různé funkce procházení adresářů a získávání souborů
- DirectoryMaker – třída pro vytváření a odstraňování adresářových struktur
- DirectoryMonitoring – třída provádějící monitorování adresáře pro změny dat
- Misc – třída poskytující různé dodatečné funkce, převážně pro rozhraní aplikace
- ServerFilesRequest – funkce požadavků k získání seznamů souborů z úložiště
- Streaming – třída zpracovává tok dat souborů získaný ze serveru
- SyncInit – inicializace synchronizace, zpracování XML a příprava stahování
- SyncCore – jádro synchronizace, stahování souborů a příprava ukládání
- UserLogin – třída ověřující autentizace a inicializuje základních objekty Google API
- XmlCreation – třída vytvářející vlastní strukturu datového XML
- XmlLoading – třída získává data z XML a zpracovává je pro třídu SyncInit

- **ConfigProcessing.dll**
  Knihovna poskytující třídy pro zpracovávání konfiguračních souborů a serializaci datových struktur pro vytvoření inicializačních souborů aplikace a dočasných souborů s informacemi o zpracovávaných souborech operacemi.
  Sestavení obsahuje třídy:
  - BinList – třída serializující seznamy s uživatelskými účty a dočasnými archívy
  - ExternalData – třída zpracovávající inicializační XML aplikace
  - Serialization – tuto třídu využívá BinList třída k serializaci

- **DataStructure.dll**
  Definování tříd vlastních datových struktur.
  Sestavení obsahuje třídy:
  - DataArray – třída, jejíž instance poskytuje pole s informacemi datového XML
  - HashMap – třída vytvářející vnitřní otisk (MD5 Hash Map) souborů
  - ListLocal – třída poskytující vlastní generický seznam
  - MimeType – třída s funkcí k získání typu souborů (MIME)
  - MyQueue – třída poskytující vlastní generickou frontu s událostí
  - Parsers – třída s různými funkcemi k manipulaci s textovými řetězci
  - XmlStruct – třída s definicí struktury datového XML využívaná třídou XmlCreation
  - ZipArguments – třída k udržení více typů argumentů předávaných jako jeden objekt

- **EncodingLibrary.dll**
  Tato knihovna poskytuje třídy nutné k zpracovávání šifrování a archivů souborů.
  Sestavení obsahuje třídy:
  - ArchiveManaging – vytváření dvou druhů archívů, šifrované a nešifrované
  - EncodingFiles – třída držící informace o zavedeném šifrování během synchronizace
  - InformationSecurity – třída poskytující šifrovací a dešifrovací funkce
  - UnicodeConversion – třída ke konverzi znaků, které nezpracuje ukládací protokol

- **UploadProcessing.dll**
  Knihovna s třídami, které provádějí inicializaci ukládání na server během samostatného ukládání ale i synchronizačního.
  Sestavení obsahuje třídy:
    - InitUpload – třída k inicializaci ukládání a zavádění různých ukládacích protokolů
    - UploaderBase – základní třída s ResumableUploader protokolem
    - UploaderFilesOnly – třída dědící od základní, souží pouze pro ukládání souborů
    - UploaderSync – třída dědící od základní, souží k synchronizačnímu ukládání souborů
    - UploaderXml – třída dědící od základní, ukládá pouze datové XML
    - UploaderXmlSync – třída dědící od základní, ukládá pouze datové XML během sync.

5.2.2 Jmenné prostory ze všech knihoven

Některé děděné třídy Google GData API mají odlišný jmenný prostor. Zde jsou uvedeny jmenné prostory, které je potřeba v aplikaci uvádět:

- Jmenné prostory základních knihoven, uvedených v kapitole 5.1.
- Vlastní knihovny, jejíž název odpovídá názvu jmenného prostoru, uvedené v kapitole 5.2.1.
- Ionic, Ionic.Zip
  Jmenný prostor pro knihovnu DotNetZip.
- Google.Documents
  Nejedná se o knihovnu Google.GData.Documents. Jmenný prostor Google.Documents poskytuje následující třídy, děděné z různých tříd Google GData API, popsané v kapitole 3.2.1:
    - Google.Documents.Document – datový typ reprezentující prvek úložiště
    - GoogleDocuments.DocumentsRequest – třída k provádění požadavků
- Google.GData.Client.ResumableUpload
  Jmenný prostor pro ResumableUploader protokol k ukládání dat na server.

5.3 Vlastnosti aplikace

Jedná se o desktopovou aplikaci, vyžadující prostředí .NET Framework 4 pro svůj běh. Aplikace je pojmenovaná jako Synchronizace s Google Drive, zkrácený název, používaný procesem běžícím na pozadí je Synchronizace s GDrive a pojmenování samotného procesu je SyncGDrive. Je umožněno zavést pouze jednu instanci aplikace.

5.3.1 Základní požadavky aplikace

Aplikace pracuje na základě vybrání adresáře aplikací, jeho uložení na úložiště a následné možnosti synchronizace takto uloženého adresáře jako kolekci. Veškeré změny provedené v lokálním adresáři se projeví změnami dat v kolekci úložiště.

Základní prvek této aplikace je datový XML soubor s informacemi o souborech synchronizovaného adresáře. Tento soubor je nutné ponechat v adresáři pro každou jeho synchronizaci. Při odstranění tohoto souboru z úložiště či lokálního adresáře, není možné provádět synchronizaci adresáře.

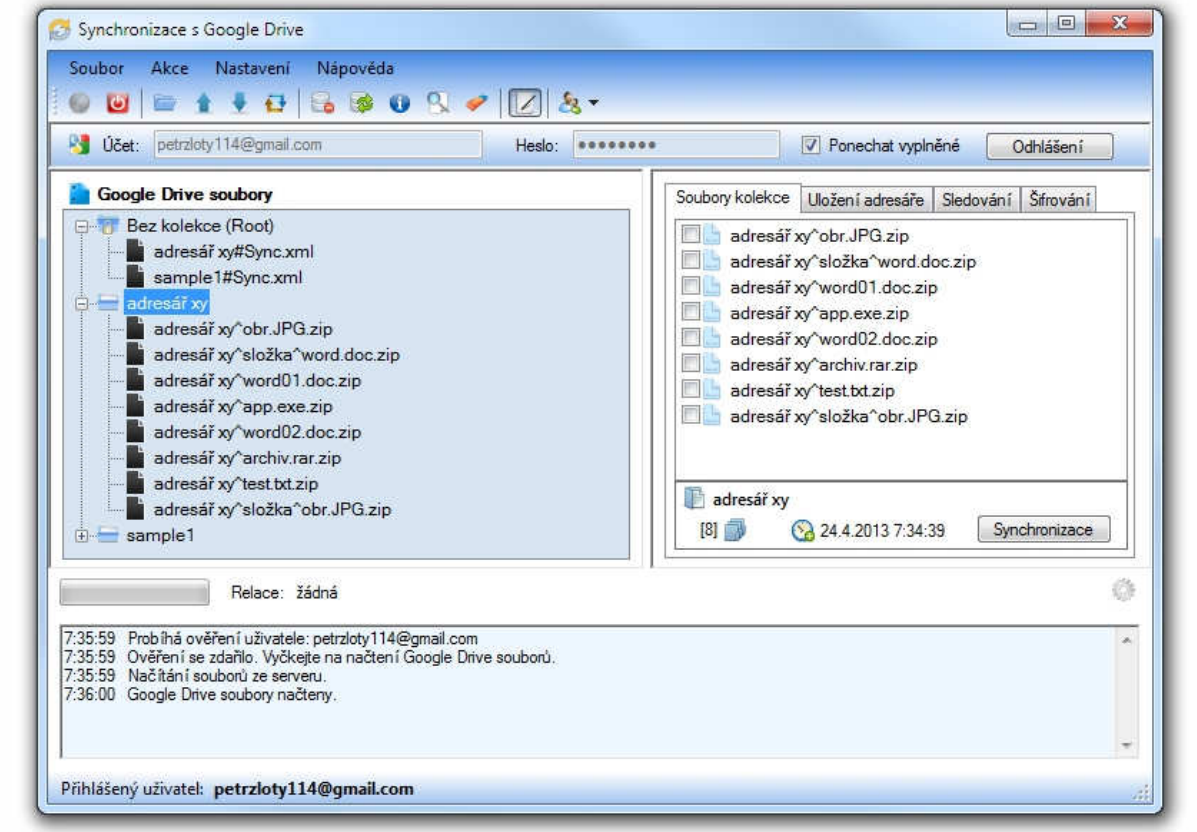
Aplikace nabízí uživateli možnost zálohovat původní adresář před synchronizací. To je provedeno vytvořením kopie adresáře ve stejné úrovni, jako je původní adresář. Po úspěšném dokončení synchronizace se záloha automaticky odstraní. V případě nečekaného přerušení operací, je možnost původní data díky této záloze obnovit, ovšem soubory úložiště není možné nahradit původními, protože není možnost tvořit zálohu souborů v kolekci.

V případě ukládání nebo stahování souborů aplikací, se musí vytvořit archívy souborů. To kvůli případům popsaným v kapitole 5.4. Archívy se vytváří v dočasném adresáři v úrovni původního. Tyto soubory se po dokončení operací odstraňují společně s dočasným adresářem.

Aplikace využívá proměnné prostředí operačního systému (Temp) k ukládání datového XML, které se aktuálně stáhlo z úložiště. Tato operace je nutná k přečtení informací o datech v úložišti. Po dokončení požadované operace se datové XML z úložiště odstraňuje, nebo se přesune do adresáře aktuálně staženého z úložiště.

Bez uvedených faktorů, aplikace nemůže provádět potřebné operace. Shrnutí požadavků aplikace, společně s identifikací faktorů, je uvedeno v následujícím seznamu:

- Datové XML aplikace odpovídající adresáři má vždy název: „název adresáře“#Sync.xml
- Záloha adresáře nese název: „název adresáře“#SyncBackup
- Dočasný adresář pro archívy souboru nese název: „název adresáře“#SyncTemp
- V proměnném prostředí (Temp) se vytváří složka „_GdocsSyncTemp“ pro stažené XML
- Archív souboru nese název relativní cesty souboru v adresáři. Důvod popsán v kapitole 5.4.



Obrázek 7: Uživatelské rozhraní aplikace

5.3.2 Charakteristické funkce aplikace

Zkráceně uvádím hlavní principy funkce aplikace před popisem její funkcionality, která je podrobně popsána v kapitole 5.5:

1) Aplikace nevyžaduje přesun souborů do zvláštní složky.
   - Vyžaduje pouze prvotní uložení adresáře na úložiště jako kolekci.
   - Každý uložený adresář získá XML soubor s informacemi o své struktuře.
   - Aplikace ale nutně vyžaduje ponechávání datového XML v adresáři.
   - Synchronizuje se celý obsah adresáře.

2) Ukládání souborů do úložiště pouze jako archívů.
   - Využití k šifrování a ochraně heslem.
   - Archivování je nutné i pro běžné soubory, důvody popsány v kapitole 5.4.

3) Šifrování a ochrana dat heslem.
   - Implementováno základní šifrování souborů k zajištění jejich nečitelnosti v úložišti.
   - Heslo se zadává pouze při archivaci, není dále nikde zaznamenáváno.

5.4 Podstatné problémy a řešení

Aplikace byla vyvíjena v době neexistence Google Drive služby, ale Google Documents. Většina zde uvedených problémů a jejích nestandartních řešení, je z důvodu implementace jádra aplikace v době fáze vývoje Google Documents List API.

Omezení vznikala s nedostatkem možností původní služby Google Documents. S dnešní službou Google Drive některá omezení odpadají, avšak s použitým Documents List API je stále nutnost využívat dále uvedených řešení. Hlavní nedostatky implementované aplikace jsou:

- Absence tvorby adresářové struktury v úložišti, řešeno pomocí archivování souborů.
- Není možné sdílení mezi více uživatelskými účty.

5.4.1 Nutnost archivování všech souborů před ukládáním

Archivování souborů je zapotřebí pro vytvoření šifrovaného souboru chráněného heslem. Nutnost archivovat i soubory nešifrované, je z následujícího důvodu. V době existence Google Documents nebylo možné tvořit v úložišti adresářové struktury, pouze kolekce. Kolekce nenahrazovaly tuto strukturu. Dále bylo zapotřebí předejít konverzi některých souborů na nativní formát Google, bylo nutné ukládat soubory v archívu. V poslední řadě, úložiště nepřijme soubory ukládané pomocí protokolu ResumableUploader v případě, kdy název souboru obsahuje Unicode znak č,ě,ď, Č,Ě,Ď.

Název archívu nese relativní cestu souboru v adresáři s nahrazenými konkrétními znaky. Tento krok vedl k zachování původního názvu souboru v archívu. Také se tímto zajistila jednoznačná identifikace souboru v úložišti, neboť uložený soubor v úložišti postrádá jakoukoliv informaci o svém původním umístění.

5.4.2 Nutnost využívání datového XML s informacemi o adresáři

V případě synchronizace adresáře s úložištěm je zapotřebí určitým způsobem zaznamenávat informace o struktuře dat tohoto adresáře. Úložiště neposkytuje možnosti získání základních vlastností uložených souborů. Pro byla volba využití XML k zápisu informací o souborech nejpraktičtější. V XML jsou uloženy základní vlastnosti souborů, jejich cesty, vnitřní otisk. Čas poslední synchronizace je nejpodstatnější a zaznamenává se v XML souboru.

5.4.3 Nemožnost plnohodnotné automatické synchronizace

Automatická synchronizace adresáře je možná pouze po jeho uložení do úložiště, nebo po synchronizaci už existující kolekce. Více o tomto způsobu v popisu funkcionality v kapitole 5.5.11.

5.5 Hlavní funkcionalita aplikace

V této kapitole jsou podrobně rozebrány hlavní implementované části, požadované zadáním práce.

5.5.1 Základy aplikační části

Hlavní informace:

- Jmenný prostor: App
- Základní formulář aplikace: TaskBarApp, pojmenování instance: appWork
- Uživatelské rozhraní: WorkWindow, pojmenování instance: appTask

Aplikace samotná běží na pozadí, její jediná vizuální komponenta je Forms.NotifyIcon, přidávající ikonu v prostoru hodin OS. Slouží k oznamovacím účelům při skrytém uživatelském rozhraní. Tato aplikace zpracovává inicializaci proměnných z inicializačního souboru aplikace při jejím zapnutí. Současně udržuje okno uživatelského rozhraní a všechny jeho instance. Instance formuláře uživatelského rozhraní se inicializuje současně s aplikací.

Uživatelské rozhraní je další formulář, informace hlavní aplikaci předává přes instance, událostmi komunikuje pouze s formuláři, které nemají inicializaci při startu aplikace. Veškeré akce na tlačítkách a nabídkách je možné provádět pouze v závislosti na konkrétní situaci, tzn. nedostupnost většiny akcí během provádění hlavních operací, aby se zabránilo nežádoucích úkonů. Je možné uživatelské rozhraní skrýt, instance jsou udržovány aplikací na pozadí.

Všechny operace se provádí ve vláknech, většina za využití komponenty BackgroundWorker. Jednak v situacích, kdy je potřeba provádět složité operace a nutno znát jejích přesné momenty ukončení s patřičným výsledkem, dále operace provádějící testování na pozadí, která je nutno zastavit v požadované momenty. Situace provádění operací na pozadí jiných operací, jsou řešeny pomocí systémových tříd Thread.

5.5.2 Přihlašovací část

Hlavní informace:

- Třída pro zpracování přihlašování: UserLogin, jmenný prostor AppProcessing

Uživatelem zadané přihlašovací údaje se ověří instancí třídy UserLogin. Při inicializaci této instance se v konstruktoru ověřují předané uživatelské parametry inicializací GData.Client.GDataCredentials a GData.Documents.DocumentsService. Instance této třídy v případě neplatného ověření vyvolá výjimku. Ta je zachycena v metodě uživatelského rozhraní, odkud se inicializuje UserLogin třída a oznamuje uživateli o nesprávných údajích.

Při ověření uživatele se dále inicializuje instance GData.Documents.DocumentsService sloužící aplikaci k provádění požadavků na úložiště a GData.Client.GDataCredentials, které slouží ResumableUploader protokolu k ukládání souborů do úložiště. Důležité je předání GData.Client.GDataCredentials objektu při inicializaci třídy GData.Client.RequestSettings, která je poté zapotřebí třídě DocumentsRequest. Popis těchto tříd v kapitole 3.2.

Po inicializaci Google GData API tříd je autentizační proces dokončen. Některé autorizační údaje jsou udržovány, protože jsou zapotřebí pro každý požadavek směrovaný na úložiště. V tento okamžik se provede aplikací požadavek na získání všech souborů z úložiště a jejich výpis do stromové struktury v okně uživatelského rozhraní.

Požadavek na získání dokumentů trvá, uživatel nemůže provádět žádné operace s daty, akce tlačítek těchto operací se neprovedou. Odhlášení v momentu získávání souborů ze úložiště je možné až po dokončení operací nebo automatickým přerušením pomocí akce odhlášení uživatele. Tyto akce jsou podrobně popsané kapitolou 5.5.3.

Odhlášení uživatele se provede za předpokladu jeho předchozího přihlášení. Zruší se instance všech tříd Google GData API, odstraní se nenaplněná kolekce z úložiště, za předpokladu nedokončeného ukládání. Nastavení výchozích hodnot proměnných a vlastností formuláře je další nezbytný krok. Před odhlašováním ještě mohou nastat situace, kdy je nutné přerušit možné probíhající operace:

- Probíhá samostatné stahování z úložiště. Zastavuje se akcí uživatele.
- Probíhá samostatné ukládání na úložiště. Zastavuje se akcí uživatele.
- Je zapnuta automatická synchronizace. Je nutné ji vypnout uživatelem.
- Získávají se soubory z úložiště. Vyčká na moment zastavení a pokračuje odhlašováním.

V případě vypnutí aplikace uživatelem během ukládání souborů na úložiště, se ukládání zruší, ale zůstávají dočasné archívy v dočasném adresáři a nekompletní kolekce v úložišti. Aplikace pro tuto situaci ukládá všechny cesty k dočasným souborům do binárního souboru. Při dalším zapnutí, aplikace navrhne uživateli odstranění všech dočasných souborů a nekompletních ukládání.

Odhlášení během synchronizace není možné, neboť synchronizaci nejde přerušit bez nechtěné ztráty dat. Vypnutím aplikace se zruší synchronizace, odhlásí se uživatel a současně zůstane záloha adresáře pro pozdější obnovu dat.

5.5.3 Požadavky k získání seznamu souborů z úložiště

Hlavní informace:

- Třída pro zpracování požadavků: ServerFilesRequest, jmenný prostor AppProcessing
- Vždy prováděno ve vlákně bwRefreshDocuments, nebo bwServerFilesTest
- Nutnost instance třídy GData.Client.RequestSettings

Požadavky na získání souborů z úložiště jsou dalším nejnutnějším prvkem aplikace. Aplikace musí vždy znát existující soubory a kolekce v úložišti jako datový typ Document. Tento požadavek ale trvá a je dále závislý na časové prodlevě mezi serverem a klientskou aplikací, je závislý na počtu dat a velikostí využitého místa v úložišti. Doba zpracování požadavků na jakékoliv počty je za běžných okolností 2s. Doba získání seznamu souborů a kolekcí se liší na jejich počtu. Celkový čas je vždy větší

než nebo roven 2s. V případě zaplněného úložiště tisícem souborů, tento požadavek může trvat i 20-30s.

Lze získat dále pouze seznam kolekcí bez souborů a seznam souborů konkrétní kolekce, přičemž je nutné znát tuto konkrétní kolekci jako typ Document. Z toho plynou jistá omezení na operace aplikace, kdy není možné získávat prvky z úložiště v reálném čase.

Prvky z úložiště získávám jako objekt GData.Client.Feed a pomocí property Entries dále získávám generickou kolekci datového typu Document, touto kolekcí iteruji a každý soubor ukládám do seznamu datového typu Document. Tento seznam, obvykle pojmenovávaný listOfAllDocs vždy obsahuje aktuální prvky úložiště reprezentované datovým typem Document. Generický seznam nabízí účelovější možnosti, než získávání generické kolekce z objektu GData.Client.Feed. Získaný seznam prvků aplikace používá k:

- Výpis dokumentů do stromové struktury komponenty TreeView.
- Získávání toku dat ze serveru k stahování souborů.
- Při synchronizaci k porovnávání existujících souborů na úložišti.
- K odstraňování kolekcí a souborů kolekce.

5.5.4 Výběr adresáře k jeho uložení na úložiště

Je zapotřebí uložit adresář na úložiště před jeho synchronizací. Tomu musí předcházet krok jeho výběru. Aplikace má implementovanou možnost přetáhnutí vybraného adresáře do svého okna, nebo se využije možnost výběru adresáře přes nabídku. Při výběru adresáře jsou implementovány následující funkce a testování:

- Otestuje se název adresáře:
    a. Nesmí se vybírat určité adresáře, jako kořenový adresář disku.
    b. V názvu adresáře nesmí existovat znaky č,ě,ď,Č,Ě,Ď, neboť se ukládá XML se stejným názvem jako adresář, ResumableUploader neuloží tyto soubory.
- Využije se test existence adresáře v úložišti jako kolekce.
  Adresář nesmí existovat vícekrát v úložišti, otestuje se podle získaného seznamu prvků jeho existence v úložišti. Aplikace vždy zabrání vytvoření takového adresáře.
- Provede se výběr všech souborů adresáře k získání jejích absolutních cest.
    a. Názvy souborů mohou obsahovat výše uvedené znaky, protože se archivují.
    b. Název souborů nesmí obsahovat znak ^, využívá se jako náhrada lomítka v názvu archívu. Archív má název jako relativní cesta k souboru.
    c. Pokud adresář obsahuje už i datové XML odpovídající jeho názvu, algoritmy výběrů souborů jej přeskočí. Později toto XML nahrazuji novým, vytvořeným před ukládáním souborů na úložiště.
- Název adresáře se dále testuje k zjištění provedení jeho minulého, kompletního vybrání.
- Samotný výběr souborů adresáře je možné zastavit, po jeho zastavení se aktuální výběrová relace zruší a uživatel může dálš vybírat.
- Průběh výběrů souborů je zaznamenáván aplikací v části uložení adresáře.
- Testování otevřených souborů v jiné aplikaci je implementováno.
- Dokončení výběru adresáře následuje vytvoření prázdné kolekce na úložišti.
    a. Kolekci se zapíše stejný název jako adresáře.

b. Provede se požadavek na získání souborů k zjištění Id této kolekce pro ukládání.

Po dokončeném výběru adresáře a vytvoření kolekce zpřístupňuji možnost ukládání souborů na úložiště. Aktuální výběr je možné zrušit a pokračovat jiným výběrem.

Hlavní informace:

- Třída CollectionManage k otestování existujících kolekcí v úložišti.
- Třída DirectoryBrowser k provedení výběrů souborů.
- Výběr souborů se provádí ve vlákně bwBrowseFolder, tvoření kolekce v bwCreateCollection.

5.5.5 Tvorba XML adresáře

Před samotným ukládáním souborů je zapotřebí jejích archivace. Současně s archivací se vytváří i XML soubor podle načtených cest získaných souborů výběrem adresáře. Struktura XML souboru je uvedena v ukázce výpisu 10. XML soubor s informacemi o souborech obsahuje element „folder“ označující adresář, dále elementy „file“ odpovídají jednotlivým souborům. Popis atributů elementů:

- name – název prvku
- selection – cesta k vybranému adresáři
- sync – čas poslední synchronizace, potřebné k zjišťování aktuálnějších synchronizací
- count – počet souborů v adresáři
- serverName – název archivovaného souboru
- path – absolutní cesta k souboru
- relative – relativní cesta k souboru bez hlavního adresáře
- mime – původní typ souboru
- serverMime – vždy typ archívu
- created – čas vytvoření souboru
- change – čas poslední změny obsahu souboru
- hashmap – vnitřní otisk souboru

```
<?xml version="1.0" encoding="windows-1250"?>
<folder name="SampleFolder01" selection="C:\SampleFolder01" sync="10.4.2013
11:30:22" size="20480" count="1">
  <file name="word.doc" serverName="SampleFolder01^word.doc.zip">
    <path>C:\SampleFolder01\word.doc</path>
    <relative>\word.doc</relative>
    <mime>application/msword</mime>
    <serverMime>application/zip</serverMime>
    <size>20480</size>
    <create>10.4.2013 0:41:32</create>
    <change>10.4.2013 11:26:12</change>
    <hashmap>B414E2F5667EB88E9A5150EB82688A6D</hashmap>
  </file>
</folder>
```

Výpis 8: Struktura aplikačního XML

XML se vytváří třídou XmlCreation, kde se zapisují patřičné informace do struktury, která je definovaná třídou XmlStruct z jmenného prostoru DataStructure. Soubor po vytvoření uzamknu, způsobem neustálého čtení FileStream souboru. Tímto způsobem zajistím jeho neodstranění z adresáře během ukládání archívů a je takto umožněno se soubory adresáře dále pracovat po vytvoření archívů.

5.5.6 Archivace souborů před ukládáním

Archivování soborů se provádí způsobem:

- Využívám ZipArguments objekt k uchování informací o následujících proměnných:
  - seznam s cestami všech souborů
  - URL připravené kolekce pro ukládání
  - cestu k vybranému adresáři
  - identifikace označující synchronizaci
- Připraví se vytvoření souboru s uloženým seznamem cest k dočasným archívům.
- Inicializuje se instance třídy InitUpload.
- Vytvořím archívy ve vlákně bwCreateArchives a předám jako argument objekt ZipArguments.
- Archivování je společné pro samotné ukládání a synchronizační ukládání, v případě synchronizace se XML nevytváří zde, ale až později.
- Využívám frontu souborů pro ukládání, naplní se každou z cest souborů.
- Ukončí se bwCreateArchives a vytvoří se seznam a soubor s cestami k archívům, v tento okamžik jsou archívy připraveny k ukládání na úložiště. Pokračuje se ihned, využije se zavedená instance InitUpload k spuštění ukládání, popsáno v kapitole 5.5.8.

5.5.7 Šifrování souborů

Šifrování souborů je možné zavést pouze před započetím archivování souborů, které je následované jejích ukládáním. Heslo se zadává před touto akcí a jeho potvrzením se pokračuje. Zrušením se ukládá bez šifrování. Zavádím objekt EncodingFiles k udržení šifrovacích údajů. Heslo se v proměnných ukládá šifrovaně. Po dokončení ukládání souborů na úložiště se provede archivace souboru XML stejným způsobem, jako běžné soubory. Uloží se na úložiště šifrované v archívu. Není běžně čitelné, jeho náhled je možný při jeho stažení a zadání hesla.

Veškeré operace, jako synchronizace a stahování, si vyžádají zadání hesla při prvotním stažení XML. Po zadání správného hesla se opět zavádí EncodingFiles objekt k držení šifrovacích údajů potřebných k šifrování nových ukládaných souborů a dešifrování stahovaných souborů. Na konci synchronizace se opět uloží šifrované XML na úložiště.

Objekt EncodingFiles se vždy ruší po dokončení operací, při zrušení operací a při vyvolání výjimek. Šifrované soubory nemají šifrované jména v archívech. Šifruje se funkcionalitou nabízenou knihovou DotNetZip při archivování souborů, využívá se WinZipAES256 algoritmu.

5.5.8 Ukládání souborů na úložiště

Ukládání na úložiště je řešeno pomocí třídy ResumableUploader z GData.Client jmenného prostoru. Nedostatky tohoto protokolu, konkrétně nemožnost ukládat konkrétní znaky v názvu, jsou uvedené v kapitole 5.4.1.

Funkcionalita ukládání se skládá z tříd jmenného prostoru aplikace UploadProcessing. Základní třída definuje ResumableUploader, ostatní třídy dědí z ní a liší se:

- Události jsou rozdílné pro samostatné ukládání nebo synchronizační ukládání.
- Rozdílné jsou URL pro ukládání XML do kořenové složky úložiště a URI potřebné pro ukládání souborů do vlastní kolekce. Kapitola 3.3.3 poskytuje hlubší popis této problematiky.

Průběh ukládání souborů na úložiště je následující:

- Inicializovanou instancí během archivování se spustí ukládání, nejprve vybráním pouze prvního prvku z fronty.
- Jakmile se uloží první soubor, událost z patřičné třídy ukládání oznámí jeho ukončení aplikaci. Poté se zavede vybírání dalšího souboru z fronty a pokračuje do posledního souboru.
- V události oznamující hlavní aplikaci o uložení souboru se provádí neustálá kontrola počtu uložených souborů. Tímto způsobem se zjišťuje ukončení ukládání všech souborů.
- Po dokončení uložení všech souborů se musí vyčkat, neboť čas vytvoření soborů v úložišti není stejný jako čas oznámení dokončení ukládání ResumableUploader protokolem.
- Po vyčkání se provede dodatečná kontrola souborů v úložišti:
    - zavede se vlákno bwServerFilesTest, test je společný i pro synchronizaci
    - odemkne se zamknuté XML v lokálním adresáři
    - inicializuje se instance třídy ServerFilesTest, získají se požadavkem soubory kolekce
    - získají se lokální soubory přečtením informací z XML
    - porovnají se lokální soubory se soubory kolekce, získají se možné neuložené soubory
    - v instanci třídy ServerFilesTest se zavede instance XmlCreation a zavolá metoda odstranění elementu „file“, který odpovídá neuloženému souboru.
- Po dodatečné kontrole se smažou dočasné archivované soubory.
- Uloží se XML na server, a to původní, nebo upravené při neuložení některých souborů.
    - v případě šifrování souborů se před uložením provede archivace a zašifrování XML.
- Událost z třídy ukládání XML souboru oznámí celkové ukončení ukládání.
- Oznámí se uživateli zpráva o ponechání XML adresáři, pokud nemá automatické nastavení ponechávání. Nastaví se výchozí stavy rozhraní aplikace a smažou se binární soubory s cestami k dočasným souborům.
- V případě zavedení monitorování adresáře se provede následující:
    - kontrola ponechání XML v adresáři, což je požadované
    - spustí se monitorování adresáře, tímto se zavádí automatická synchronizace
- Zruší se objekt držící šifrovací údaje, v případě zavedeného šifrování.

Během ukládání souborů na úložiště je možné s původními soubory adresáře pracovat díky ukládání jejích archívů. Nesmí se ale odstranit XML z adresáře, proto je zavedeno jeho zamknutí touto aplikací po dobu ukládání souborů.

Zastavení ukládání na úložiště je možné pouze během samostatného ukládání bez synchronizace. Dočasné soubory se odstraní, kolekce se odstraní a zruší se veškeré zavedené instance.

V případě neuložení některých souborů, při ukládání jejich velkých množství, se situace napraví výše uvedenou kontrolou uložených souborů. Každou následnou synchronizací adresáře se tyto neuložené soubory uloží. Tento aspekt se naskytne při nedostatku místa v úložišti.

5.5.9 **Synchronizace**

Samotná synchronizace se provádí akcí uživatele. Tento způsob synchronizace není automatický při změnách dat v adresáři, ale právě volbou uživatele. Při zavedení synchronizace se provede běžným způsobem synchronizace dat adresáře a kolekce na úložišti. Po jejím dokončení, jsou obě místa synchronizované. Další synchronizace se provádí opět akcí uživatele. Automatickou synchronizaci nahrazuje volba monitorování adresáře v kapitole 5.5.11. Pro zavedení synchronizace je zapotřebí:

1) Označit kolekci ve stromové struktuře na rozhraní aplikace a zvolit synchronizaci.

2) Musí existovat lokální adresář odpovídající kolekci, pokud neexistuje, je možné provést:
   - Kompletní stažení do původní cesty podle kolekce.
   - Stažení do jiné cesty, vždy ale musí uživatel označit adresář s názvem shodujícím se s názvem kolekce. Změna cesty adresáře se změní i v XML.

3) Adresář existuje, ale nemá XML se svou strukturou. Tady se odstraňují soubory adresáře a stáhnou se soubory kolekce. Vždy je nutné mít XML v adresáři, bez jeho existence není možné synchronizovat žádný adresář.

4) Uživatel musí uzavřít soubory adresáře při počátečních testech, aplikace dále upozorňuje na otevřené soubory. Po testech je možné zase pracovat se soubory, ale změny jsou zaznamenány pouze v případě zavedené automatické synchronizace.

5) Je možné mít otevřené soubory v aplikacích, které nepřidávají zámek na otevřené soubory. Takové soubory aplikace neodhalí, je zapotřebí, aby uživatel uložil tyto otevřené soubory před synchronizací.

6) Synchronizaci nelze přerušit bez ztráty dat. Náhrady jsou možné pouze z vytvářené zálohy.

Třídy, poskytující veškeré operace využívané u synchronizace:

- ServerFilesRequest
- SyncInit
- XmlLoading
- DataArray
- Backup
- SyncCore
- ArchiveManaging
- DirectoryBrowser
- CollectionManage
- ZipArguments
- InitUpload
- XmlCreation
- UploaderXmlSync

Celý proces synchronizace se skládá z testů souborů a operací stahování a ukládání. Nejprve se testují soubory potřebné ke stahování z úložiště a poté soubory potřebné k uložení na úložiště. Pokaždé v tomto pořadí, tomu následují i operace. Celý postup implementované synchronizace je následující:

1. Ze seznamu souborů úložiště se vyhledá XML odpovídající označené kolekci.

2. Provede se stažení XML do složky v proměnném prostředí OS pomocí třídy SyncInit. Šifrované XML vyžaduje zadání hesla, ověřením hesla se pokračuje jeho dešifrování.

3. Otestuje se šifrování a zavádí se objekt držící šifrovací informace a smaže se šifrované XML.

4. Zavede se čtení získaného XML souboru, inicializují se seznamy pro testované soubory. Dále se provádí test, k zjištění otevřených souborů a poskytuje se oznámení o této situaci.

5. Třídou XmlLoading se provede načtení informací serverových souborů.

6. Pokud k serverovému souboru existuje soubor v lokálním adresáři, otestuje se vnitřní otisk souborů. V případě rozdílného otisku, se provede porovnání času poslední změny:
   - Serverový je novější – serverový soubor se stáhne a nahradí lokální.
   - Lokální je novější – nebude se provádět stahování souboru.

7. Lokální soubor z bodu 6. neexistuje, existuje pouze serverový, ověří se následující:
   - XML v adresáři neexistuje, stáhnou se všechny soubory ze serveru.
   - XML v adresáři existuje, následují testy uvedené v bodě 8.

8. Zjistí se odstranění serverových souborů, podle novějšího času poslední synchronizace XML na serveru a XML v adresáři:
   - Novější synchronizaci má serverové XML – později bude stažen serverový soubor.
   - Novější synchronizaci má lokální XML – později bude odstraněn serverový soubor.

9. Vytvoří se potřebná adresářová struktura pro stahované soubory.

10. Využije se objekt DataArray k uchování všech přečtených informací o souborech z XML, dále se využije tento objekt k předávání jako jediný argument třídě aplikace.

11. Pokud k lokálnímu souboru neexistuje serverový soubor, provedou se testy pro zjištění nutnosti ukládat nebo odstranit přebývající soubor lokálního adresáře oproti serverové kolekci:
    - Pokud je čas poslední synchronizace novější na serveru, smaže se lokální soubor, který je navíc v adresáři v porovnání se serverovou kolekcí.

12. Provede se stažení potřebných serverových souborů z úložiště a zpracování předcházejících testů k odstraňování souborů. Vše probíhá následovně:
    - Ověří se povolené zálohování lokálního adresáře.
    - Inicializuje se instance SyncCore, vytvoří se dočasný adresář pro stahování souborů.

- Vytvoří se tok dat podle souborů z úložiště datového typu Documents.
- Začne zpracovávání získaného toku dat, stahuje se soubor do dočasného adresáře.
- Po stažení všech souborů se provede jejich extrakce z archívů, příprava cest a souborů se provádí v SyncCore třídě, ale samotná extrakce až v třídě ArchiveManaging.
- Rozlišuje se šifrování inicializovaným objektem EncodingFiles.

13. Provedou se testy lokálních souborů k ukládání do úložiště, běží ve vlákně stahování souborů.
    - Vyhledají se nově přidané lokální soubory, které neexistují v úložišti ani v adresářovém XML. Budou se později ukládat do úložiště.
    - Vyhledají se serverové soubory, které je nutné smazat z úložiště – neexistují už v lokálním adresáři. Za předpokladu aktuálnějšího XML adresáře.
    - Vyhledají se soubory, které je nutné uložit do úložiště – jsou novější v adresáři.
    - K odstranění ze serveru se přidá i XML. Po každé synchronizaci nechávám uložit nové XML adresáře, i když se nic nestahuje a neukládá.

14. Po dokončení stahování souborů, se provede ukládání souborů podle provedených testů.
    - Využívá se stejné funkcionality třídy InitUpload, která se využívá při archivaci souborů před zavedením ukládání.
    - Zde se ale nevytváří XML adresáře, jako je tomu v případě samostatného ukládání.
    - XML adresáře se vytváří ihned po dokončení stahování souborů, současně s ukládáním. Tento XML soubor není nikdy součástí seznamů s testovanými soubory.
    - Může nastat situace, kdy není nutné ukládat soubory – následuje bod 16.
    - Po archivaci souborů, proběhne jejich uložení na úložiště stejným způsobem, jako u samostatného ukládání, s rozdílem využívání tříd UploaderSync a UploaderXmlSync.

15. Ukončení operací synchronizace je vždy zaváděno událostmi, oznamujícími ukončení ukládání z tříd ukládání souborů během synchronizace – UploaderSync a UploaderXmlSync.

16. Při ukončení synchronizace se provádějí další potřebné operace.
    - Odstraní se prázdné adresáře.
    - Provede se dodatečný test existence souborů na úložišti – stejný test, jako po samostatném ukládání souborů, popřípadě se upraví XML adresáře o neuložené soubory.
    - Uloží se XML, v případě zavedeného šifrování se archivuje – stejný postup jako při samostatném ukládání.
    - Třída pro ukládání XML souborů oznámí jeho kompletní uložení a v tento okamžik je synchronizace kompletní. Následují akce v bodě 17.

17. Synchronizace je kompletní, provedou se pouze operace k zavedení výchozích stavů aplikace:
    - Výchozí stavy komponent uživatelského rozhraní.
    - Požadavek k získání souborů z úložiště a jejich vypsání do stromové struktury.
    - Odstranění binárního souboru s cestami k archivovaným souborům.
    - Odstraní se vytvořená záloha původního adresáře.
    - Zruší se objekt držící šifrovaní údaje – pouze v případě zavedeného šifrování.

- Testuje se zavedení monitorování adresáře. Pokud je uživatelem zavedeno, spustí se automatická synchronizace adresáře, který se aktuálně synchronizoval.

Během provádění funkcionality synchronizace mohou nastat situace, závislé na aspektech:

- Při odstraňování souborů během synchronizace může být konkrétní soubor otevřený jiným procesem – aplikace upozorní na tento výskyt a nepokračuje dále, dokud uživatel neuzavře soubor.

- Změny pouze malá-velká písmena původních znaků v názvu souborů se neprojeví okamžitým ukládáním tohoto souboru, ale zápisem této změny v adresářovém XML, které se ukládá na úložiště. V úložišti zůstává původní název, avšak při každém stažení souboru se mu zapíše změněný znak názvu.

5.5.10 Samostatné stahování souborů bez synchronizace

V aplikaci je implementována možnost stahovat soubory z úložiště bez synchronizace. Tato vlastnost je výhodná v situaci neexistence adresáře na disku uživatele. Synchronizace ale v tomto případě pracuje stejným způsobem, jediný rozdíl je v neaktualizování XML a jeho znovu-ukládání během samostatného stahování souborů.

Samostatné stahování pouze stáhne obsah označené kolekce a na konci se dotáže na přesun XML z úložiště do tohoto adresáře. Vytvoří se adresář na disku uživatele, připravený k následné synchronizaci. Nutno poznamenat hlavní aspekty funkcionality:

- Situace, kdy adresář existuje, nahradí se veškeré jeho soubory těmi z úložiště.
- Stahování je možno zastavit, oproti synchronizaci.

Funkcionalita stahování je pouze využití částí implementace synchronizace. Využívá se stejných metod, pouze s rozdílnými parametry. Rozdíly jsou následující:

- neprovádí se žádné testy souborů
- při čtení XML se získá pouze název souborů v úložišti
- při stahování souborů se oznamuje aplikaci průběh jinými událostmi, oproti synchronizaci
- po ukončení stahování se nepokračuje, jako při synchronizaci
- nevytváří se XML k uložení na úložiště

5.5.11 Automatická synchronizace

Automatickou synchronizaci nahrazuje v aplikaci implementovaná možnost sledování adresáře.
V jiných nástrojích se provádí automatická synchronizace speciální složky nástroje. Aplikace nabízí možnost synchronizace kteréhokoliv adresáře, uloženého jako kolekci v úložišti. Není možné zavést automatickou synchronizaci všech adresářů, uložených jako kolekci s implementovanou funkcionalitou jádra synchronizace. Automaticky se synchronizuje vždy pouze jedna kolekce.

Rozdíl, oproti automatické synchronizaci speciální složky, je v principu synchronizace, implementované v této aplikaci. Nemožnosti plnohodnotné automatické synchronizace jsou v následujících faktorech této implementace:

- Jádro synchronizace umožňuje synchronizaci volbou, kterou uživatel zavede a ukončuje se po provedení všech operací.
- Neimplementoval jsem neustálé čtení XML v reálném čase a sledování adresáře v jádru synchronizace. Jedině tímto způsobem by se dosáhlo plnohodnotné automatické synchronizace.

Jako náhradu automatické synchronizace jsem implementoval sledování změn v adresáři s využitím funkcí stávající synchronizace. Ovšem tento krok má jistá omezení:

1) Není možné funkcionalitou synchronizace využívat konkrétní informace, získávané ze sledování změn adresáře systémovou třídou. Důvody popsány níže.

2) V případě změny v datech sledovaného adresáře se pouze spustí funkcionalita implementované synchronizace. Spustí se ale i v případě otevření souboru beze změny jeho obsahu.

3) Je nutné zavřít všechny otevřené soubory, které mají zámek od jiné aplikace, až poté se provede synchronizace. Během čekání na uzavření konkrétního souboru se neprovádí.

Funkcionalitu sledování změn adresáře poskytuje třída DirectoryMonitoring. Třída využívá systémovou třídu FileSystemWatcher k zjišťování změn a předávání pomocí událostí. Události této třídy poskytují název změněného, odstraněného a vytvořeného souboru. V případě otevření konkrétních formátů např. dokumentů v aplikaci MS Word dochází k vytvoření několika dočasných souborů této aplikace. FileSystemWatcher upozorní ale i na tyto dočasné skryté soubory a neposkytne název původního otevřeného dokumentu v aplikaci MS Word. Zde bylo nutné spoléhat na vlastní vyhledávání otevřených souborů a pouze tímto způsobem získávat tyto konkrétní uzamknuté soubory.

Druhým aspektem FileSytemWatcher funkcí je předávání informací o změněných datech několikrát pro jeden prvek. To z důvodu způsobu hledání změn touto třídou, třída nabízí pouze dvě události oznamované při změně, vytvoření, smazání a přejmenování souboru. Vždy se generují vícenásobné události při jedné změně souboru, to díky odlišným způsobům práce různých aplikací s daty, při jejich zápisu, ukládání, otevírání dat.

Vlastní testy adresáře k vyhledávání otevřených souborů musí při každé objevené změně třídou FileSystemWatcher prohledávat adresář a zjišťují otevření souborů pomocí čtení Stream souboru. Tato akce je náročnější na čas při výskytu velkých souborů. Tímto odpadá efektivní testování v reálném čase a implementace tohoto způsobu do stávající funkcionality synchronizace.

Protože není možné získat třídou FileSytemWatcher název některých otevřených souborů, bylo nutné funkcionalitu sledování adresáře rozšířit o vlastní způsoby:

- Ve vlákně komponenty BackgroundWorker se zavádí instance třídy FileSytemWatcher.
- Provádí se v cyklu testování k zastavení BackgroundWorker komponenty a provádí se testování fronty, která je plněná soubory, získanými pomocí událostí třídy FileSystemWatcher.
- Pokud je fronta naplněná soubory, provede se rekurzivně vyhledání otevřených souborů.
- Při nálezu otevřeného souboru se postupuje dále:
    - existence souboru – kvůli výskytu skrytých dočasných souborů
    - vynucení k zavření souboru uživatelem pomocí informativní zprávy
    - čeká se na uzavření souboru jeho testováním v cyklu
    - oznámení o uzavření souboru

    - ve frontě mohou existovat dočasné skryté soubory, které ale na disku po uzavření souboru už neexistují – provede se test existence těchto souborů a odstranění z fronty
    - tyto akce se provedou pro každý soubor adresáře
- Po akcích v předchozím bodě je nutné otestovat frontu, zdali nedošlo k přidání dalších souborů.
    - odstraní se z fronty neexistující dočasný soubory aplikace, jako např. MS Word
    - odstraní se z fronty už uzavřený soubor, který je vždy několikrát evidován díky aspektům, uvedeným v popisu třídy FileSystemWatcher
- BackgroudWorker komponenta se ukončuje v případě neexistence dalšího oznámeného souboru ve frontě. V tomto okamžiku není žádný soubor otevřen. Testování se ukončuje.
- Aplikace oznámí uživateli o zavádění synchronizace.
    - podle monitorovaného adresáře se vyhledá kolekce
    - spustí se implementovaná synchronizace aplikace
- Po dokončení synchronizace se opět zavede sledování adresáře, pokud je povoleno uživatelem.

Zrušením monitorování adresáře se zastaví testování a neprovede se synchronizace. Další monitorování je možné zavést po synchronizaci adresáře uživatelem. Během synchronizace je možné mít otevřené soubory ve většině aplikací, které neposkytují zámek na zpracovávané soubory. Sledování adresáře vždy upozorní na uložení takto otevřeného souboru a ihned provádí synchronizaci.

5.6 Vedlejší funkcionalita aplikace

5.6.1 Náhledy XML souborů z úložiště

Aby bylo možné zjistit informace o uloženém adresáři jako kolekci v úložišti, aplikace poskytuje možnost náhledu XML souboru. Náhled umožňuje zobrazit informace z XML souboru jako seznam a jako jeho vlastní strukturu. Využívá se části funkcionality synchronizace, kdy se provede stažení a následné čtení obsahu XML souboru k zobrazení uživateli. Po uzavření náhledu se XML odstraní.

5.6.2 Stahování a odstraňování pouze označených souborů

Tato funkcionalita byla implementovaná pro případy, kdy uživatel potřebuje stažení pouze konkrétního souboru z kolekce. Nevyužívá se zde funkcionality synchronizace, nestahuje se XML odpovídající kolekce. Soubory se ukládají jako archívy do uživatelem zvoleného adresáře. Další extrakce už provádí uživatel.

5.6.3 Účty pro rychlé přihlašování

Aplikace nabízí dynamicky vytvářený seznam uživatelských účtu k rychlému přihlašování. Tato možnost je řešená způsobem, kdy si uživatel definuje v sekci nastavení své uživatelské účty, které se uloží do binárního souboru users.bin v adresáři aplikace. Při vytváření není nutné ukládat heslo, pouze účet. Pokud se uživatel rozhodne k uložení hesla, je zapsáno šifrovaně. Binární soubor zůstává pouze v pracovním adresáři aplikace.

Při zapnutí aplikace se vždy načítá tento soubor a dynamicky se vytváří seznam v poli nabídek uživatelského rozhraní. Uživatel si vybere, který uložený účet se mu vypíše do přihlašovacích formulářů. Tato funkce umožní rychlé zadávání uživatelských údajů, obzvláště ocenitelné při používání více účtů k službám Google, stejně tak při vývoji a neustálém testování aplikace.

6 Závěrečné zhodnocení

Implementovaná aplikace synchronizuje uživatelem zvolený adresář mezi více počítači za využívání úložiště Google Drive.

Hlavní funkcionalita aplikace byla vyvíjena v době služby Google Documents s využíváním dnes již neaktuálního Google GData API. Původní služba Google Documents neumožňovala vytváření adresářové struktury v úložišti, proto bylo nutné využívat datového XML s informacemi o struktuře a datech uloženého adresáře. Nutnost archivování souborů má původ v problémech ukládajícího protokolu GData Documents List API, ale také k zajištění jednoznačné identifikace archívu podle názvu, bez jakékoliv změny původního souboru. Způsoby získávání seznamů dokumentů z úložiště jako jeden celek neumožňovaly efektivní získávání informací z úložiště v reálném čase, neboť tento požadavek trvá delší dobu v různých závislostech. Proto je automatická synchronizace řešená nestandardním způsobem. Tomu přispívají i nedostatky sledování změn v adresáři, při využívání systémového oznamování změn adresáře.

Využívané GData Documents List API je možné migrovat na Google Drive API, avšak tento krok může vyžadovat kompletní přepis funkcionality, v lepším případě pouze částí a následné testování, neboť verze API pro .NET a dokumentace jsou stále ve fázi vývoje.

GData Documents List API je možné využívat, ačkoliv už není aktuální. Nutno podotknout, že s tímto API bylo dosaženo požadované funkcionality, i když s jistým omezením. Rozšíření uvedených omezení je s přechodem na Google Drive API možné, rozhodně nabízí možnosti adresářové tvorby a bezproblémové ukládací protokoly.

Při srovnání aplikace s existujícími nástroji, je možné sledovat výhody navrženého šifrování a ochraně dat heslem. Zde další vývoj vidím v jiných možnostech využívání šifrovacího algoritmu a klíče. Klíč je součástí třídy poskytující šifrování a ta je součástí knihoven aplikace. V neposlední řadě je výhoda v nepotřebě přesouvat soubory k synchronizaci. Umožnil jsem synchronizovat libovolný adresář formou jeho uložení na úložiště a uchováváním datového XML v adresáři ke každé synchronizaci. XML je identifikováno specifickým názvem a jeho uchovávání v adresáři není omezující. Tento způsob ale není vlastní většině oficiálních nástrojů.

Synchronizace je navržena pouze jako uživatelem spustitelná akce a to z výše uvedených důvodů, stejně tak automatická synchronizace. Obě operace i přesto mohou provádět požadovanou funkcionalitu ze zadání práce a to synchronizaci adresáře uživatele mezi více osobními počítači.

S využitím Google Drive API by mohla odpadnout nutnost využívání datového XML adresáře, díky možnostem tvořit adresářovou strukturu v úložišti. Ovšem je nutné, aby API poskytovalo možnosti získání vnitřního otisku uložených souborů a přístup k jejich časům poslední změny obsahu. Dále je nutné určitým způsobem uchovávat v kolekci informaci o lokálním umístění uloženého adresáře a jeho času poslední synchronizace. S těmito faktory se stále jeví nutnost využívat určitý souboru s informacemi, ukládaného současně s adresářem na úložiště.

Reference

[1] TUHÝ, Radan. Svět hardware, [online], 2013. Úložiště dat na internetu: k datům odkudkoliv. Dostupné z WWW: http://www.svethardware.cz/uloziste-dat-na-internetu-k-datum-odkudkoli/36307.

[2] BARTOLŠIC, Martin. Computerworld, [online], 2012. Je výhodné mít data v cloudu? Dostupné z WWW: http://computerworld.cz/technologie/je-vyhodne-mit-data-v-cloudu-48581.

[3] KLIMÁNEK, Oldřich. DSL.cz, [online], 2011. Také se bojíte se soubory v DropBoxu?. Dostupné z WWW: http://www.dsl.cz/clanek/2358-take-se-bojite-se-soubory-v-dropboxu-zkuste-wuala-s-lepsim-zabezpecenim.

[4] Microsoft Corporation, Microsoft Windows, [online]. Smlouva o poskytování služeb společnosti Microsoft. Dostupné z WWW: http://windows.microsoft.com/cs-cz/windows-live/microsoft-services-agreement.

[5] Google Inc., Google Apps, [online].
Dostupné z WWW:http://www.google.com/enterprise/apps/business/index.html.

[6] Dropbox Inc., Dropbox Developers, [online].
Dostupné z WWW:https://www.dropbox.com/developers.

[7] Microsoft Coorporation, MSDN, [online]. Dostupné z WWW: http://msdn.microsoft.com.

[8] Google Inc., Google Developers, [Online], Google Documents List API, 2012.
Dostupné z WWW: https://developers.google.com/google-apps/documents-list/.

[9] Google Inc., Google Code, [Online], .NET library for the Google Data API, 2012.
Dostupné z WWW: https://code.google.com/p/google-gdata/.

Adresářová struktura přiloženého média

- Application Zkompilované soubory aplikace
- Source Zdrojové kódy aplikace
- Doc Soubor práce, zadání, abstrakt a klíčová slova